# 3.2. Árboles AVL

## DEFINICIONES (I)

- La eficiencia en la búsqueda de un elemento en un árbol binario de búsqueda se mide en términos de:
  - Número de comparaciones
  - La altura del árbol
- Árbol completamente equilibrado: los elementos del árbol deben estar repartidos en igual número entre el subárbol izquierdo y el derecho, de tal forma que la diferencia en número de nodos entre ambos subárboles sea como mucho 1
- Problema: el mantenimiento del árbol
- Árboles AVL: desarrollado por Adelson-Velskii y Landis (1962). Los AVL son árboles balanceados (equilibrados) con respecto a la altura de los subárboles:

  *"Un árbol está equilibrado respecto a la altura si y solo si para cada uno de sus nodos ocurre que las alturas de los dos subárboles difieren como mucho en 1"*
- Consecuencia 1. Un árbol vacío está equilibrado con respecto a la altura
- Consecuencia 2. El árbol equilibrado óptimo será aquél que cumple:

  $n = 2^h - 1$, donde $n$ = nº nodos y $h$ = altura

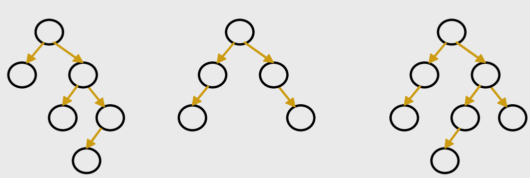




# 3.2. Árboles AVL

## DEFINICIONES (II)

- Si T es un árbol binario no vacío con TL y TR como subárboles izquierdo y derecho respectivamente, entonces T está balanceado con respecto a la altura si y solo si
  - TL y TR son balanceados respecto a la altura, y
  - $| hl - hr | \leq 1$ donde hl y hr son las alturas respectivas de TL y TR
- El **factor de equilibrio FE** ( T ) de un nodo T en un árbol binario se define como hr - hl. Para cualquier nodo T en un árbol AVL, se cumple FE ( T ) = -1, 0, 1



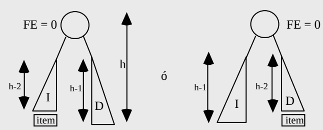
# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. INSERCIÓN (I)

- **Representación de árboles AVL**
  - Mantener la información sobre el equilibrio de forma implícita en la estructura del árbol
  - Atribuir a, y almacenar con, cada nodo el factor de equilibrio de forma explícita

```
TNodoArb {
        Titem fitem;
        TArbBin fiz, fde;
        int FE;   }
```

- **Inserción en árboles AVL. Casos:**
  - Después de la inserción del ítem, los subárboles I y D igualarán sus alturas



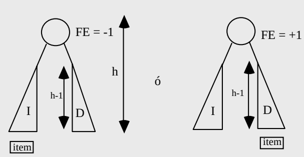





# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. INSERCIÓN (II)

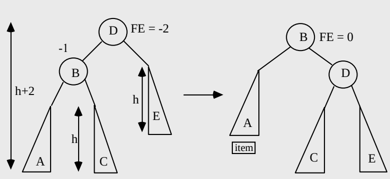
- Después de la inserción, I y D tendrán distinta altura, pero sin vulnerar la condición de equilibrio




- Si hI > hD y se realiza inserción en I, ó hI < hD y se realiza inserción en D

  Formas de rotación: II, ID, DI, DD

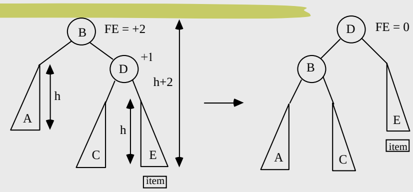
- ROTACIÓN II **(-2,-1)**



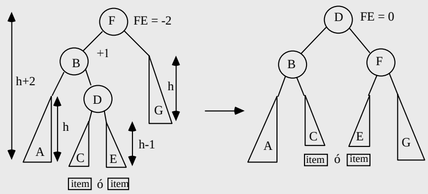
# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. INSERCIÓN (III)

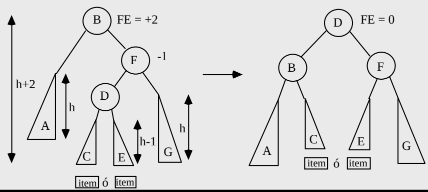
- ROTACIÓN DD
  **(+2,+1)**




- ROTACIÓN ID
  **(-2,+1)**




- ROTACIÓN DI
  **(+2,-1)**





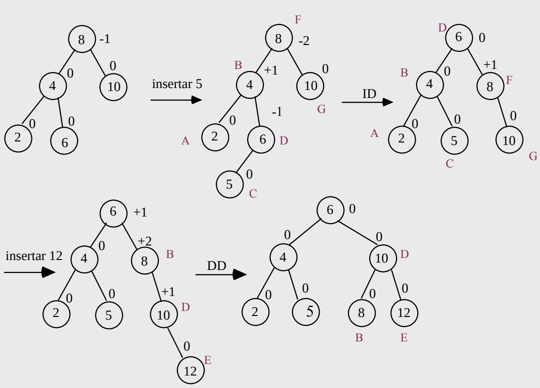




# 3.2. Árboles AVL


## OPERACIONES BÁSICAS. INSERCIÓN. EJEMPLO (IV)

- **Ejemplo**. Insertar en el siguiente árbol los elementos 5 y 12




- Hay que tener en cuenta que la actualización del FE de cada nodo se efectúa desde las hojas hacia la raíz del árbol

# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. INSERCIÓN. IMPLEMENTACIÓN (V)

```
ALGORITMO INSERTAR
    ENTRADA/SALIDA
            A: AVL; c : Item
    VAR I : Iterador ; Crece : Integer ;
    METODO
            I = Primer ( A ) ;
            InsertarAux ( I, c, Crece ) ;
    fMETODO
```

```
ALGORITMO INSERTARAUX
    ENTRADA/SALIDA I : Iterador;   Crece: Integer; c : Item ;
    VAR CreceIz, CreceDe : Integer ; B : Arbol ;
    METODO
            si EsVacioArbIt ( I ) entonces
                B = Enraizar ( c ) ; Mover ( I, B ) ; Crece = TRUE ;
            sino
                Crece = CreceIz = CreceDe = FALSE ;
                si ( c < Obtener ( I ) ) entonces
                    INSERTARAUX ( HijoIzq ( I ), c, CreceIz ) ;
                    Crece = CreceIz ;
                sino
                    si ( c > Obtener ( I ) ) entonces
                        INSERTARAUX ( HijoDer ( I ), c, CreceDe ) ;
                        Crece = CreceDe ;
                    fsi
                fsi
                si Crece entonces
                    caso de:
                        1) ( CreceIz y FE ( I ) = 1 ) ó ( CreceDe y FE ( I ) = -1 ) :
                                Crece = FALSE ; FE ( I ) = 0 ;
                        2) CreceIz y FE ( I ) = 0 : FE ( I ) = -1 ;
                        3) CreceDe y FE ( I ) = 0 : FE ( I ) = 1 ;
                        4) CreceIz y FE ( I ) = -1 : EquilibrarIzquierda ( I, Crece ) ;
                        5 ) CreceDe y FE ( I ) = 1 : EquilibrarDerecha ( I, Crece ) ;
                    fcaso
                fsi
            fsi
    fMETODO
```



© DLSI (Univ. Alicante) Tema 3. Tipo árbol

# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. INSERCIÓN. IMPLEMENTACIÓN (VI)

```
ALGORITMO EQUILIBRARIZQUIERDA
    ENTRADA/SALIDA I : Iterador;   Crece: Integer;
    VAR J, K: Iterador; int E2;
    METODO
            si ( FE (HijoIzq (I ) = -1 entonces              //ROTACIÓN II
                        Mover (J, HijoIzq (I));
                        Mover (HijoIzq (I), HijoDer (J));
                        Mover (HijoDer (J), I);
                        FE (J) = 0; FE (HijoDer (J)) = 0;
                        Mover (I,J);
            sino                                             //ROTACIÓN ID
                        Mover (J, HijoIzq (I));
                        Mover (K, HijoDer (J));
                        E2 = FE (K);
                        Mover (HijoIzq (I), HijoDer (K));
                        Mover (HijoDer (J), HijoIzq (K));
                        Mover (HijoIzq (K), J);
                        Mover (HijoDer (K), I);
                        FE (K) = 0;

                        caso de E2
                            -1: FE (HijoIzq (K)) = 0; FE (HijoDer (K)) = 1;
                            +1: FE (HijoIzq (K)) = -1; FE (HijoDer (K)) = 0;
                            0: FE (HijoIzq (K)) = 0; FE (HijoDer (K)) = 0;
                        fcaso
                        Mover (I, K);
            fsi
            Crece = FALSE;
    fMETODO
```

# 3.2. Árboles AVL

## EJERCICIOS *inserción*

1) Construir un árbol AVL formado por los nodos insertados en el siguiente orden con etiquetas 4, 5, 7, 2, 1, 3, 6

2) Insertar las mismas etiquetas con el siguiente orden: 1, 2, 3, 4, 5, 6, 7

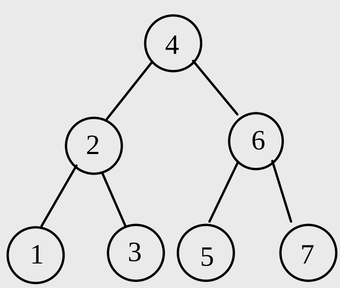




# 3.2. Árboles AVL

## EJERCICIOS *inserción*: SOLUCIÓN

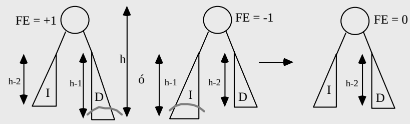
1) La solución para los 2 ejercicios es la siguiente:



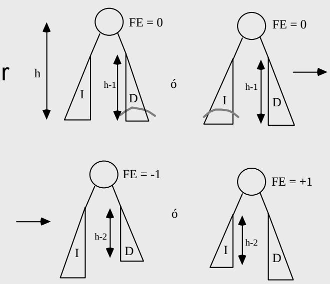
# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. BORRADO (I)

- **Borrado en árboles AVL. Casos:**
  - Borrar el ítem nos llevará en el árbol a un FE = 0, no será necesario reequilibrar




  - Borrar el ítem nos llevará en el árbol a un FE = ±1, en este caso tampoco será necesario reequilibrar



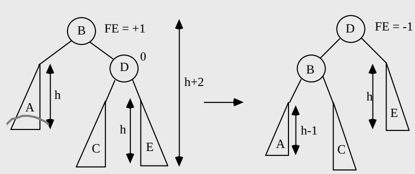




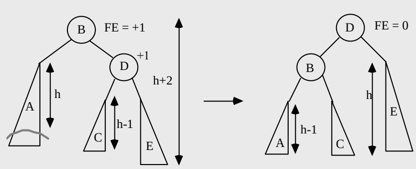

# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. BORRADO (II)

- Rotaciones simples
  - ROTACIÓN DD
    **(+2,0)**



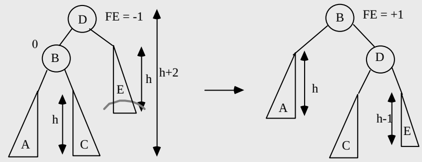

**(+2,+1)**
**La altura del árbol decrece**



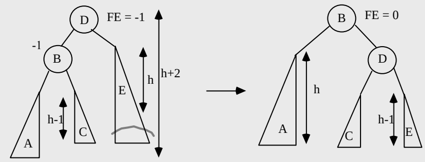
# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. BORRADO (III)

- Rotaciones simples
  - ROTACIÓN II
    **(-2,0)**




**(-2,-1)**
**La altura del árbol decrece**



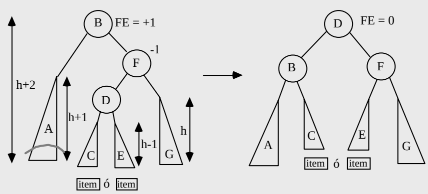




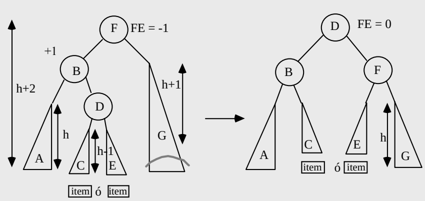

# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. BORRADO (IV)

- Rotaciones dobles
  - ROTACIÓN DI
    **(+2,-1)**
    **La altura del árbol decrece**




  - ROTACIÓN ID
    **(-2,+1)**
    **La altura del árbol decrece**



# 3.2. Árboles AVL

## OPERACIONES BÁSICAS. INSERCIÓN Y BORRADO

- **Estudio de las complejidades de ambos algoritmos**
  - El análisis matemático del algoritmo de inserción es un problema todavía no resuelto. Los ensayos empíricos apoyan la conjetura de que la altura esperada para el árbol AVL de *n* nodos es

    h = log2 ( *n* ) + c / c es una constante pequeña
  - Estos árboles deben utilizarse sólo si las recuperaciones de información (búsquedas) son considerablemente más frecuentes que las inserciones → debido a la complejidad de las operac. de equilibrado
  - Se puede borrar un elemento en un árbol equilibrado con log ( *n* ) operaciones ( en el caso más desfavorable )
- **Diferencias operacionales de borrado e inserción:**
  - Al realizar una inserción de una sola clave se puede producir como máximo una rotación ( de dos o tres nodos )
  - El borrado puede requerir una rotac. en todos los nodos del camino de búsqueda
  - Los análisis empíricos dan como resultado que, mientras se presenta una rotación por cada dos inserciones,
  - sólo se necesita una por cada cinco borrados. El borrado en árboles equilibrados es, pues, tan sencillo ( o tan complicado ) como la inserción

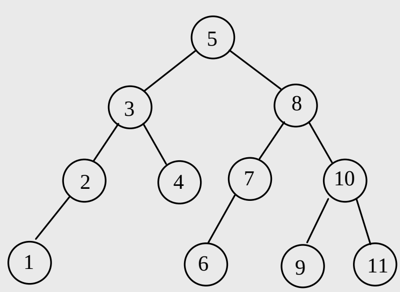




# 3.2. Árboles AVL

## EJERCICIOS *borrado*

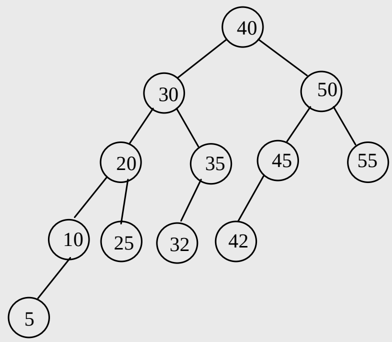
1) Dado el siguiente árbol AVL de entrada, efectuar los siguientes borrados en el mismo: 4, 8, 6, 5, 2, 1, 7. (Nota: al borrar un nodo con 2 hijos, sustituir por el mayor de la izquierda)



# 3.2. Árboles AVL

## EJERCICIOS *borrado*

2) Dado el siguiente árbol AVL de entrada, efectuar los siguientes borrados en el mismo: 55, 32, 40, 30. (Nota: al borrar un nodo con 2 hijos, sustituir por el mayor de la izquierda)








# 3.2. Árboles AVL

## EJERCICIOS *borrado*: SOLUCIÓN

2) La solución es la siguiente: